AMENDMENT TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims.

Listing of Claims

1. (Currently Amended) A memory, comprising:

a memory array having a plurality of storage elements;

a plurality of replacement storage elements;

a plurality of address fuse units, each having a plurality of fusible links and being operable to store a replacement address, each replacement address identifying one of the storage elements of the memory array to be replaced by an associated one of the replacement storage elements and forming a respective ~~2^m bit~~ row or ~~2^n bit~~ column of a fuse array having 2^m bit rows and 2^n bit columns;

a vector generator operable to produce a 2^n bit row vector based on the 2^m bit rows of the fuse array and to produce a 2^m bit column vector based on the 2^n bit columns of the fuse array; and

a compression unit operable to produce a row checksum from the row vector and to produce a column checksum from the column vector.

2. (Currently Amended) The memory of claim [2] 1, wherein: the vector generator is operable to produce each bit of the row vector by determining a logic combination of the 2^m bits of a corresponding one of the rows of the fuse array; and the

vector generator is operable to produce each bit of the column vector by determining a logic combination of the 2^n bits of a corresponding one of the columns of the fuse array.

3. (Original) The memory of claim 2, wherein: the vector generator is operable to produce each bit of the row vector by determining an XOR of adjacent ones of the 2^m bits of the corresponding row of the fuse array; and the vector generator is operable to produce each bit of the column vector by determining an XOR of adjacent ones of the 2^n bits of the corresponding column of the fuse array.

4. (Original) The memory of claim 3, wherein: the 2^m bits of the corresponding row of the fuse array are sequentially bit 1, bit 2, . . . bit 2^m and the vector generator is operable to produce each bit of the row vector by determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a (2^{m-1})th XOR of bit 2^m and a previous XOR; the 2^n bits of the corresponding column of the fuse array are sequentially bit 1, bit 2, . . . bit 2^n and the vector generator is operable to produce each bit of the column vector by determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a (2^{n-1})th XOR of bit 2^n and a previous XOR.

5. (Original) The memory of claim 1, wherein: the compression unit is operable to produce each bit of the row checksum by determining a logic combination of at least some of the 2^n bits of the row vector; and the compression unit is operable to produce each bit of the column checksum by determining a logic combination of at least some of the 2^m bits of the column vector.

6. (Original) The memory of claim 5, wherein: the compression unit is operable to produce each bit of the row checksum by determining an XOR combination of at least some of the 2^n bits of the row vector; and the compression unit is operable to produce each bit of the column checksum by determining an XOR combination of at least some of the 2^m bits of the column vector.

7. (Original) The memory of claim 6, wherein: each bit of the row vector is represented by an address of n bits, An, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the row vector is capable of representation by address An=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one; the compression unit is operable to produce an ith bit of the row checksum by determining a logic combination of an ith set of bits of the row vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . n-1, and j=1, 2, 3, . . . n; and the compression unit is operable to produce a kth bit of the row checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . n.

8. (Original) The memory of claim 7, wherein the compression unit is operable to produce each ith bit and each kth bit of the row checksum by determining an XOR of the respective ith and kth sets of bits of the row vector.

9. (Original) The memory of claim 8, wherein the bits within each of the respective ith and kth sets of bits are sequentially bit 1, bit 2, . . . bit 2^{n-1} and the compression unit is operable to produce each ith bit and each kth bit of the row checksum by determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a $(2^{n-1}-1)$th XOR of bit 2^{n-1} and a previous XOR.

10. (Original) The memory of claim 6, wherein: each bit of the column vector is represented by an address of m bits, Am, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the column vector is capable of representation by address Am=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one; the compression unit is operable to produce an ith bit of the column checksum by determining a logic combination of an ith set of bits of the column vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . m-1, and j=1, 2, 3, . . . m; and the compression unit is operable to produce a kth bit of the column checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . m.

11. (Original) The memory of claim 10, wherein the compression unit is operable to produce each ith bit and each kth bit of the column checksum by determining an XOR of the respective ith and kth sets of bits of the column.

12. (Original) The memory of claim 11, wherein the bits within each of the respective ith and kth sets of bits are sequentially bit 1, bit 2, . . . bit 2^{n-1} and the compression unit is operable to produce each ith bit and each kth bit of the column checksum by determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a $(2^{n-1}-1)$th XOR of bit 2^{n-1} and a previous XOR.

13. (Original) The memory of claim 1, further comprising at least one storage unit operable to store the row checksum and the column checksum.

14. (Original) The memory of claim 13, wherein the at least one storage unit includes a plurality of fusible links to store the row checksum and the column checksum.

15. (Original) The memory of claim 13, further comprising an error detector operable to compare the stored row checksum and a subsequently determined row checksum for a difference therebetween, and to compare the stored column checksum and a subsequently determined column checksum for a difference therebetween.

16. (Original) The memory of claim 15, wherein the error detector is further operable to determine which of the replacement addresses is faulty based on at least one of the difference between the stored row checksum and the subsequent row checksum, and the difference between the stored column checksum and the subsequent column checksum.

17. (Original) The memory of claim 16, wherein: each bit of the row vector is represented by an address of n bits, An, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the row vector is capable of representation by address An=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one; the compression unit is operable to produce an ith bit of the row checksum by determining a logic combination of an ith set of bits of the row vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . n-1, and j=1, 2, 3, n; and the compression unit is operable to produce a kth bit of the row checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . n.

18. (Original) The memory of claim 17, wherein: the error detector is operable to determine faulty ones of the ith and kth bits of the subsequently determined row checksum based on the difference between the stored row checksum and the subsequently determined row checksum; the error detector is operable to determine one of the addresses of n bits contributing to all of the ith and kth sets of bits that correspond to the faulty bits of the subsequently determined row checksum; and the error detector is operable to determine that the faulty replacement address is one that corresponds to the bit of the row vector that is represented by the determined address of n bits.

19. (Original) The memory of claim 16, wherein: each bit of the column vector is represented by an address of m bits, Am, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the column vector is capable of representation by address Am=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one; the compression unit is operable to produce an ith bit of the column checksum by determining a logic combination of an ith set of bits of the column vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . m-1, and j=1, 2, 3, . . . m; and the compression unit is operable to produce a kth bit of the column checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . m.

20. (Original) The memory of claim 19, wherein: the error detector is operable to determine faulty ones of the ith and kth bits of the subsequently determined column checksum based on the difference between the stored column checksum and the subsequently determined column checksum; the error detector is operable to determine one of the addresses of m bits contributing to all of the ith and kth sets of bits that correspond to the faulty bits of the subsequently determined column checksum; and the error detector is operable to determine that the faulty replacement address is one that corresponds to the bit of the column vector that is represented by the determined address of m bits.

21. (Currently Amended) A method, comprising:

forming a fuse array from a plurality of replacement addresses, each replacement address identifying one of a plurality of storage elements of a memory array to be replaced by a replacement storage element, each replacement address forming a respective ~~2^m bit~~ row or ~~2^n bit~~ column of the fuse array having 2^m bit rows and 2^n bit columns;

producing a 2^n bit row vector based on the 2^m bit rows of the fuse array;

producing a 2^m bit column vector based on the 2^n bit columns of the fuse array;

producing a row checksum from the row vector; and

producing a column checksum from the column vector.

22. (Original) The method of claim 21, further comprising:

producing each bit of the row vector by determining a logic combination of the 2m bits of a corresponding one of the rows of the fuse array; and

producing each bit of the column vector by determining a logic combination of the 2n bits of a corresponding one of the columns of the fuse array.

23. (Original) The method of claim 22, further comprising:

producing each bit of the row vector by determining an XOR of adjacent ones of the 2m bits of the corresponding row of the fuse array; and

producing each bit of the column vector by determining an XOR of adjacent ones of the 2n bits of the corresponding column of the fuse array.

24. (Original) The method of claim 23, wherein: the 2^m bits of the corresponding row of the fuse array are sequentially bit 1, bit 2, . . . bit 2^m and the step of producing each bit of the row vector includes determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a (2^{m-1}) th XOR of bit 2^m and a previous XOR; the 2^n bits of the corresponding column of the fuse array are sequentially bit 1, bit 2, . . . bit 2^n and the step of producing each bit of the column vector includes determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a (2^{n-1}) th XOR of bit 2^n and a previous XOR.

25. (Original) The method of claim 21, further comprising: producing each bit of the row checksum by determining a logic combination of at least some of the 2^n bits of the row vector; and producing each bit of the column checksum by determining a logic combination of at least some of the 2^m bits of the column vector.

26. (Original) The method of claim 25, wherein: the step of producing each bit of the row checksum includes determining an XOR combination of at least some of the 2^n bits of the row vector; and the step of producing each bit of the column checksum includes determining an XOR combination of at least some of the 2^m bits of the column vector.

27. (Original) The method of claim 26, wherein each bit of the row vector is represented by an address of n bits, An, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the row vector is capable of representation by address An=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one, and the method further comprises: producing an ith bit of the row checksum includes determining a logic combination of an ith set of bits of the row vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . n-1, and j=1, 2, 3, . . . n; and producing a kth bit of the row checksum includes determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . n.

28. (Original) The method of claim 27, wherein the steps of producing each ith bit and each kth bit of the row checksum includes determining an XOR of the respective ith and kth sets of bits of the row vector.

29. (Currently Amended) The method of claim 28, wherein the bits within each of the respective ith and kth sets of bits are sequentially bit 1, bit 2, . . . bit 2^{n-1} and the steps of producing each ith bit and each kth bit of the row checksum includes determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a $(2^{n-1}-1)$th XOR of bit 2^{n-1} and a previous XOR.

30. (Original) The method of claim 26, wherein each bit of the column vector is represented by an address of m bits , Am, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the column vector is capable of representation by address Am=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one, and the method further comprises: producing an ith bit of the column checksum by determining a logic combination of an ith set of bits of the column vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . m-1, and j=1, 2, 3, . . . n; and producing a kth bit of the column checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . m.

31. (Original) The method of claim 30, wherein the steps of producing each ith bit and each kth bit of the column checksum includes determining an XOR of the respective ith and kth sets of bits of the column.

32. (Original) The method of claim 31, wherein the bits within each of the respective ith and kth sets of bits are sequentially bit 1, bit 2, . . . bit 2^{n-1} and the steps of producing each ith bit and each kth bit of the column checksum includes determining a first XOR of bits 1 and 2, a second XOR of a next bit and the first XOR, . . . and a $(2^{n-1}-1)$th XOR of bit 2^{n-1} and a previous XOR.

33. (Original) The method of claim 21, further comprising storing the row checksum and the column checksum.

34. (Original) The method of claim 33, further comprising producing a subsequent row checksum and a subsequent column checksum.

35. (Original) The method of claim 34, further comprising: determining whether there is a difference between the stored row checksum and the subsequent row checksum; and determining whether there is a difference between the stored column checksum and the subsequent column checksum.

36. (Original) The method of claim 35, further comprising: determining which of the replacement addresses are faulty based on at least one of the difference between the stored row checksum and the subsequent row checksum, and the difference between the stored column checksum and the subsequent column checksum.

37. (Original) The method of claim 36, wherein each bit of the row vector is represented by an address of n bits, An, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the row vector is capable of representation by address An=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one, the method further comprising: producing an ith bit of the row checksum by determining a logic combination of an ith set of bits of the row vector for which the respective Aj address bits are 0, where i=1, 3, 5, ... n-1, and j=1, 2, 3, . . . n; and producing a kth bit of the row checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . n.

38. (Original) The method of claim 37, further comprising: determining faulty ones of the ith and kth bits of the subsequent row checksum based on the difference between the stored row checksum and the subsequent row checksum; determining one of the addresses of n bits contributing to all of the ith and kth sets of bits that correspond to the faulty bits of the subsequent row checksum; and determining that the faulty replacement address is one that corresponds to the bit of the row vector that is represented by the determined address of n bits.

39. (Original) The method of claim 36, wherein each bit of the column vector is represented by an address of m bits, Am, . . . A2, A1, one of a least significant bit (LSB) and a most significant bit (MSB) of the column vector is capable of representation by address Am=0, . . . A2=0, A1=0, and each sequential bit is capable of representation by an address incremented by one, the method further comprising: producing an ith bit of the column checksum by determining a logic combination of an ith set of bits of the column vector for which the respective Aj address bits are 0, where i=1, 3, 5, . . . m-1, and j=1, 2, 3, . . . m; and producing a kth bit of the column checksum by determining a logic combination of a kth set of bits of the row vector for which the respective Aj address bits are 1, where k=2, 4, 6, . . . m.

40. (Original) The method of claim 39, further comprising: determining faulty ones of the ith and kth bits of the subsequent column checksum based on the difference between the stored column checksum and the subsequent column checksum; determining one of the addresses of m bits contributing to all of the ith and kth sets of bits that correspond to the faulty bits of the subsequently determined column checksum; and determining that the faulty replacement address is one that corresponds to the bit of the column vector that is represented by the determined address of m bits.